

Web-ohjelmointi - Muutama eimerkkiharjoitustyö

Esimerkkiharjoitustyö 1 - Erinomainen

- Ohessa **osa** harjoitustyön dokumentaatiosta kuvaruutukaappauksina
- Harjoitustyö on erityisesti tekniseltä tasoltaan erinomaista tälle opintojaksolle

Suomen avainluvut kartalla

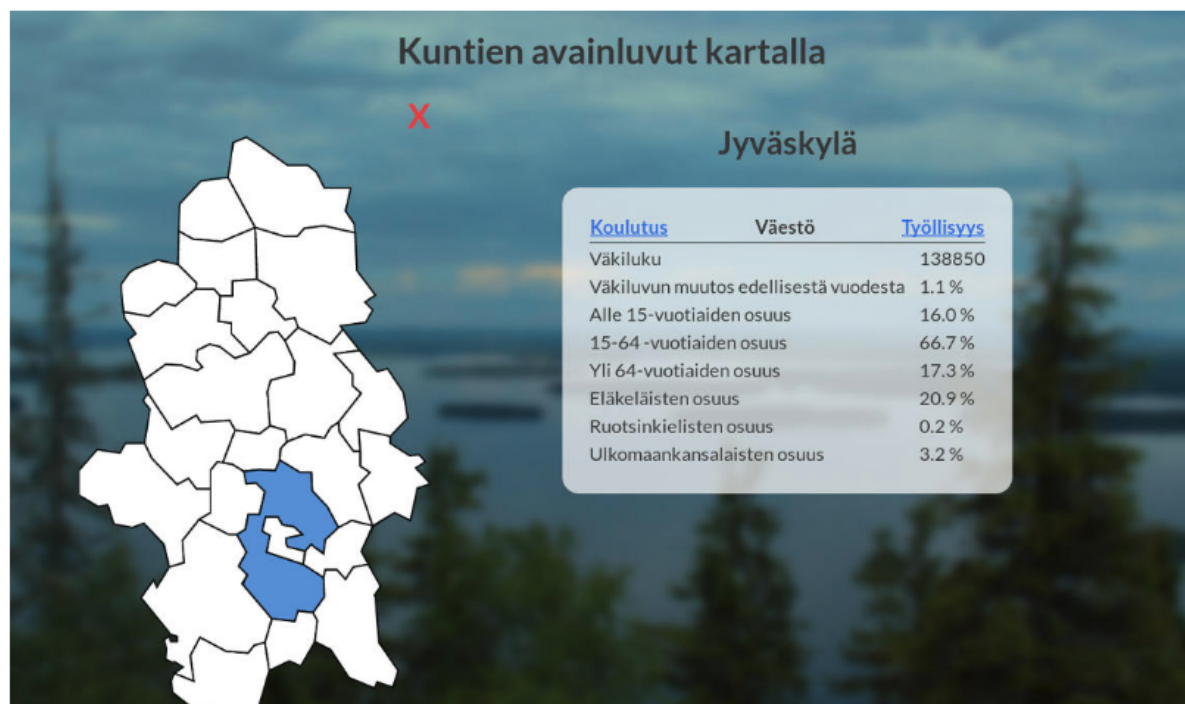
30.11.2017

TMS0500 web-ohjelmointi

Sovellus on nähtävillä DigitalOcean:in palvelimella osoitteella [redacted]

Tämä repositorio zip-paketissa: [http://student.labranet.jamk.fi/~\[redacted\]](http://student.labranet.jamk.fi/~[redacted]) zip

Yleiskuvaus



Kuntakartta-sovelluksella käyttäjä voi hiirellä klikkaamalla valita niin maakunta kuin kuntatasolla tarkasteltavan kohteen. Klikattaessa kohdetta, kohteen tiedot haetaan dynaamisesti palvelimelta. Valitun kohteen avainluvut esitetään kartan vierellä taulukossa. Taulukosta käyttäjä voi tarkastella lukuja jaoteltuina kategorioihin koulutus, väestö ja työllisyys.

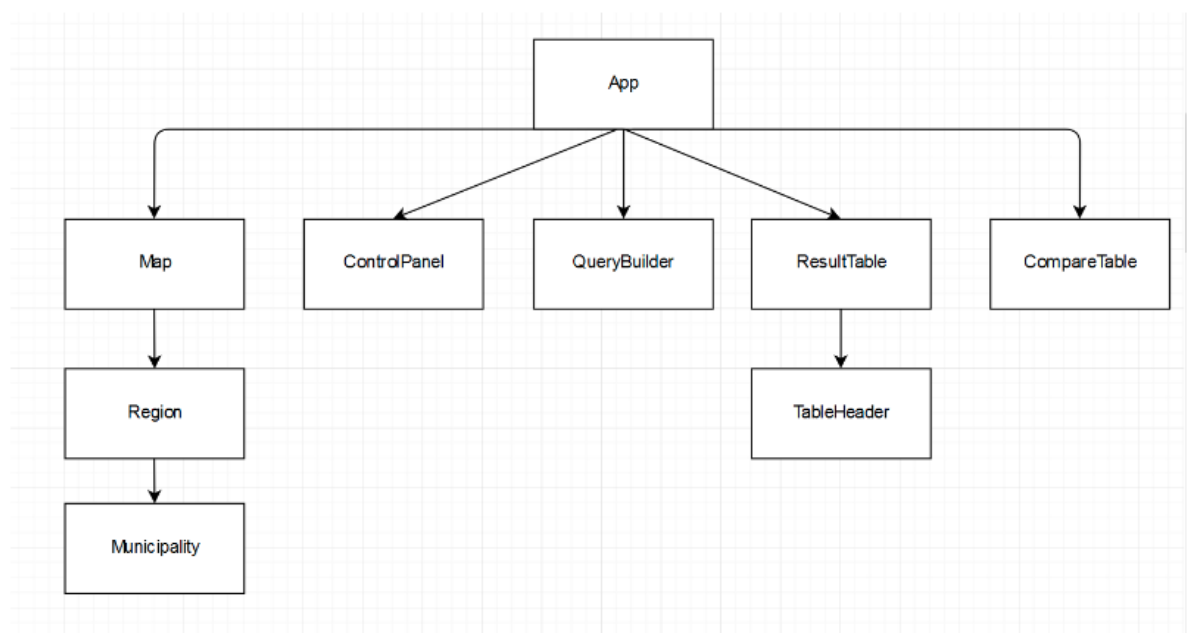
Käyttöliittymästä löytyy nappeja, joilla käyttäjä voi suorittaa muutamia ennalta määritettyjä kyselyitä, kuten "Väkiluvulta suurimmat kunnat" ja "Huonoin työllisyysaste". Näiden kyselyiden tulokset väritetään kartalle ja tulokset esitetään vertailussa kartan vierellä.

Käyttäjän on myös mahdollista luoda rajatuin ehdoin omia kyselyitä, joiden tulokset esitetään yllä kuvutulla tavalla.

Esitettävä data on haettu tilastokeskuksen avoimesta rajapinnasta csv-muodossa, josta rakensin tätä sovellusta varten relaatiotietokannan.

Käyttöliittymä on toteutettu React-kirjastolla ja se löytyy frontend kansioista. Palvelintoteutus on toteutettu Node.js + express.js pohjalta ja se löytyy backend kansioista. Tietovarastona toimii PostgreSQL tietokanta. Tietokannan luontitiedosto init.sql löytyy repositorion juuresta.

Sovelluksen rakenne



Sovelluksen kokoa App-komponentti. Tämä komponentti sisältää sovelluksen tilan state-muuttujassa sekä huolehtii ajax-kutsujen suorittamisesta ja niistä saadun datan välittämisestä alemmille komponenteille esitettäväksi. Alemmille komponenteille välitetään takaisinkutsufunktioita, joiden avulla tieto käyttäjän toiminnoista välitetään ylös App-komponentille.

- Map-komponentti hyödyntää Region- sekä Municipality komponentteja kartan piirtämiseen.
- ControlPanel sisältää nappeja, joilla voidaan suorittaa ennaltamääritettyjä kyselyitä.
- QueryBuilder sisältää lomakkeen, jolla käyttäjä voi rakentaa rajatuin ehdoin omia kyselyitään.
- ResultTable esittää valitun kohteen tiedot kategorioihin jaetussa taulukossa.
- CompareTable esittää tiedot, kun suoritetaan kysely, joka voi palauttaa useamman kuin yhden tuloksen.

Kartan piirto

Map-komponentti hoitaa kartan piirtämisen. Kartta piirretään svg-elementtinä. Path-elementtien d-attribuutti määrittää polun piirtämiseen käytetyn tiedon. Nämä tiedot luetaan data.json tiedostosta. Map-komponentti luo svg-elementin ja luo sen sisälle Region-komponentit (maakunnat).

```
render() {
  const {data, selectedItems} = this.props;

  const regions = data.map(region => {
    return <Region
      selectedItems={selectedItems}
      region={region}
      key={region.id}
      onMunicipalityClick={(id) => this.props.onMunicipalityClick(id)}
      onClick={(id) => this.props.onRegionClick(id)}
      selectionMode={this.props.selectionMode} />
  });

  ...

  return (
    <div className="map-container">
      {this.props.selectedRegion &&
        <span id="region-close" onClick={this.props.onRegionCloseClick}>X</span>}

      <svg
        xmlns="http://www.w3.org/2000/svg"
        viewBox={viewBox}>
        {this.props.selectedRegion ? this.selectedRegion() : regions}
      </svg>
    </div>
  );
}
```

Region-komponentit luovat g-elementin ja sen sisälle omat rajansa path-elementtinä sekä kunnat Municipality-komponentteina.

```
const municipalities = region.kunnat.map(m => {
  let selected = false;
  if ((selectedItems && selectedItems.indexOf(m.id) !== -1)) {
    selected = true;
  }
  return <Municipality
    selected={selected}
    d={m.d}
    id={m.id}
    key={m.id}
    onClick={() => props.onMunicipalityClick(id)}
    selectionMode={props.selectionMode} />
});

...

return (
  <g className={className} onClick={() => props.onClick(region.id)}>
    <path d={region.d} className="region-path"/>
    {municipalities}
  </g>
);
```

Municipality-komponentin tehtäväksi jää ainoastaan piirtää omat kuntarajansa path-elementtinä.

```
return (
  <path
    d={d}
    className={className}
    onClick={() => props.onClick(id)}>
    <title>{id}</title>
  </path>
);
```

Itsearvio

Ehdotan harjoitustyölle arvosanaa 5.

Olen työn lopputulokseen erittäin tyytyväinen ja toteutus onnistui hieman odotettua helpommin. Aikaa toteutukseen käytettiin noin 34 tuntia. Erityisesti kartan toiminnallisuuden toteutus onnistui jouhevasti, sillä svg:n sisältämät elementit tukevat hyvin osoittimen tapahtumia. Myös React rakentaa elementit nopeasti luetun datan mukaan ja piirto on kaikin puolin sulavaa.

Lähdekoodin rakenne on mielestäni selkeä ja hyvin jaoteltu. Uusien ominaisuuksien kuten kunnan tai maakunnan haku nimellä olisi nopea lisätä nykyiseen toteutukseen.

Käyttöliittymän reponsiivisuuteen en ehtinyt paneutua. Käyttöliittymä ei nykyisellään ole pienellä näytölle käytettävä.

Asennus

Tämän projektin kehitysympäristön asennus onnistuu helposti, mikäli sinulla on asennettuna docker sekä docker-compose. Huomaa kuitenkin, että ympäristön asennus vie useamman sataa megatavua levytilaa. Projekti on luotu create-react-app työkalulla, joka asentaa node_modules kansioon kaikenlaista ylimääräistä ~200MB arvosta. Myös PostgreSQL kontti yksissään on ~300MB. Käyttöliittymä ja palvelin käyttävät mhart/alpine-node konttia, joka on 66,3MB.

Kloonaa repositorio, siirry hakemiston juureen ja suorita docker-compose.yml komennolla

```
docker-compose up
```

docker-compose rakentaa kolme konttia: yhden käyttöliittymälle, yhden palvelimelle sekä yhden tietokannalle. Kontit käynnistyvät automaattisesti, jonka jälkeen voit siirtyä selaimella osoitteeseen localhost:3000.



Muutokset frontend/src sekä backend/src kansioiden tiedostoihin päivittävät sovellusta välittömästi tallennuksen jälkeen ilman tarvetta käynnistää sovellusta uudelleen.

Asennetut kontit voi poistaa komennolla

```
docker-compose down --rmi all
```

Esimerkkiharjoitustyö 2 - Hyvä perustyö

- Ohessa kuvaruutukaappaus tavallisen harjoitustyön käyttöliittymästä
- Käyttäjä voi katsella eri ravintoloiden ruokalistoja valitsemalleen päivämäärälle ja ravintolalle. Ruokalistojen sisältö haetaan ravintoloiden itsensä tarjoamista avoimista JSON/XML-muotoisista rajapinnoista AJAX-tekniikalla.

		Maanantai 28.5	Tiistai 29.5	Keskiviikko 30.5	Torstai 31.5
<h2>Aimo</h2>  <p>4.17/5 rating Lue arvosteluja</p>		<h2>Bittipannu</h2>  <p>2.86/5 rating Lue arvosteluja</p>			
<p>KASVISLOUNAS 8,30€/2,60€</p> <p>Kikhernepataa (* ,A ,G ,L ,M ,Veg ,VS) Höyrytettyä tummaa riisiä (* ,G ,L ,M ,Veg)</p>		<p>MIX & MATCH</p> <p>Mix & Match Buffet Poimi lautaselle mitä haluat lounaslinjast</p>			
<p>LOUNAS II 8,30€/2,60</p> <p>Juustoista jauheliha-pastapannua (A ,L ,VS)</p>		<p>VEGAANI</p> <p>Tofu-ratatouillea ja täysijyväriisiä</p>			
<p>FRESH BUFFET 17,40€/kg</p> <p>Sitrusmarinoituja katkarapuja (A ,G ,L ,M)</p>		<p>POPULAR</p> <p>Jauhelihalasagnettea</p>			
<p>LOUNAS III 9,20€/4,95€</p> <p>Paistettu broilerileike (* ,A ,L ,M) Choronkastiketta (A ,G) Rosmariini-lohkoperunoita (* ,G ,L ,M ,Veg ,VS)</p>		<p>SPECIAL LUNCH</p> <p>Piccata Milanese porsaanleike, punaviinikastiketta ja perunaa</p>			
		<p>SOUP</p> <p>Pinaattikeittoa ja keitetty kananmuna</p>			

Esimerkkiharjoitustyö 3 - Minimityö

- Ohessa kuvaruutukaappauksena **osa** harjoitustyön dokumentaatiosta, jolla on tavoiteltu vain jonkin verran arvosanan nostavaa vaikutusta opintojaksolle
- Työssä on ohjelmoitu hyvin minimaalinen selaimen lisäosa (plugin) JavaScriptillä. Dokumentaatiossa on esitetty ytimekkäästi, mikä on selaimen lisäosa, kuinka se tehdään ja kuinka se otetaan käyttöön ja halutessa jaetaan. Esimerkkinä on plugin, joka käytössä ollessaan, muuttaa "hauskasti" iltalehdistön uutisotiskoita. Ohjelmakoodia koko toteutuksessa on vähän.

```
{
  "manifest_version": 2,
  "name": "UutisStna",
  "version": "0.1",

  "description": "Uutiset hauskoiksi",

  "content_scripts": [
    {
      "matches": ["*://*.is.fi/"],
      "js": ["sites/IS/is-frontpage.js"]
    },
    {
      "matches": ["*://*.is.fi/*"],
      "js": ["sites/IS/is-article.js"]
    },
    {
      "matches": ["*://*.iltalehti.fi/*"],
      "js": ["sites/iltalehti.js"]
    }
  ]
}
```

Kuva[2]

```
var texts = document.getElementById("container_keski").getElementsByClassName("df-article-conte
for(var i = 0; i < texts.length; i++){
  try {
    texts[i].getElementsByTagName("h2")[0].getElementsByTagName("a")[0].textContent += " se
  }
  catch(asdf) {
    console.log("getElement Failed");
  }
}
```

5 Lisäosan asentaminen selaimeen ilman lataamista sovelluskaupasta

Jätetty tarkoituksella tyhjäksi